MOCSA: multiobjective optimization by conformational space annealing

Sangjin Sim, Juyong Lee and Jooyoung Lee[a]

[a]*School of Computational Sciences, Korea Institute for Advanced Study, Seoul, Korea*

Abstract

We introduce a novel multiobjective optimization algorithm based on the conformational space annealing (CSA) algorithm, MOCSA. It has three characteristic features: (a) Dominance relationship and distance between solutions in the objective space are used as the fitness measure, (b) update rules are based on the fitness as well as the distance between solutions in the decision space and (c) it uses a constrained local minimizer. We have tested MOCSA on 12 test problems, consisting of ZDT and DTLZ test suites. Benchmark results show that solutions obtained by MOCSA are closer to the Pareto front and covers a wider range of the objective space than those by the elitist non-dominated sorting genetic system (NSGA2).

Keywords: conformational space annealing, multiobjective optimization, genetic algorithm, evolutionary algorithm, Pareto front

1. Introduction

The multiobjective optimization problem (MOOP) is to optimize two or more objective functions simultaneously, subject to given constraints. The multiobjective optimization can be applied to problems where the final decision should be made considering two or more conflicting objectives. MOOP occurs in various fields such as industrial design, finance, management and many engineering areas. Practical goals in these fields can be generalized in such a way that the cost of a process is minimized while the quality of its product is maximized. The primary goal is to find a set of solutions that any individual objective function cannot be improved without deteriorating the other objective functions, and such a set is called a Pareto set. For efficient decision making, a set of generated solutions ($\mathcal{GS}$) should meet two

conditions: It should be as close to the Pareto front as possible and the solutions should be distributed as widely as possible.

Evolutionary algorithm (EA) is one of the most popular and successful approaches to solve MOOPs (Deb, 2001; Coello Coello et al., 2007). A number of EA-based algorithms have been suggested including the vector evaluated genetic algorithm (VEGA) (Schaffer, 1985), the niched Pareto genetic algorithm (NPGA) (Horn et al., 1994), the nondominated sorting genetic algorithm II (NSGA2) (Deb et al., 2000), the strength Pareto evolutionary algorithm II (SPEA2) (Zitzler et al., 2001), the mimetic Pareto archived evolution strategy (M-PAES) (Knowles and Corne, 2000) and micro genetic algorithm (micro-GA) (Coello Coello and Toscano Pulido, 2001). Among them, NSGA2 and SPEA2 are arguably the most widely used methods. Other approaches include simulated annealing (SA) (Suman and Kumar, 2005; Nam and Park, 2000), tabu search (Hansen, 1997; Gandibleux et al., 1997), particle swarm optimization (PSO) (Parsopoulos and Vrahatis, 2002; Coello Coello and Lechuga, 2002), immune algorithm (IA) (Coello Coello and Cortés, 2002; Gao and Wang, 2011; Luh et al., 2003), ant system (Doerner et al., 2004; Barán and Schaerer, 2003) and cultural algorithm (Coello Coello and Becerra, 2003).

Conformational Space Annealing (CSA) is a highly efficient single-objective global optimization algorithm which incorporates advantages of genetic algorithm and SA. It has been successfully applied to diverse single-objective optimization problems in physics and biology, such as protein structure modeling (Lee et al., 1999; Pillardy et al., 2001; Liwo et al., 1999; Joo et al., 2009), finding the minimum energy solution of a Lenard-Jones cluster (Lee et al., 2003), multiple sequence alignment (Joo et al., 2008) and the community detection problem (Lee et al., 2012) on networks. In these studies, CSA is shown to perform more efficient sampling using less computational resources than the conventional Monte-Carlo (MC) and SA methods.

Here, we introduce a new multiobjective optimization algorithm by using CSA, MOCSA. Compared to existing EAs, MOCSA has the following distinct features: (a) The ranking system considers the dominance relationship and the distance between solutions in the objective space, (b) solutions are updated by using a dynamically varying distance cutoff measure to control the diversity of the sampling in the decision space, and (c) a gradient-based constrained minimizer is utilized for local search.

The remainder of this paper is organized as follows. In section 2, the definition of MOOP and related terms are described. In section 3, details

of MOCSA is presented. Numerical results and the comparison between MOCSA and NSGA2 on various test problems are presented in section 4. The final section contains the conclusion.

2. Problem statement

The mathematical definition of a MOOP can be defined as follows,

$$\begin{aligned} &\min_{\mathbf{u}} \mathbf{v} = \min_{\mathbf{u}} \mathbf{f}(\mathbf{u}) = \min_{\mathbf{u}} \left(f_1(\mathbf{u}), f_2(\mathbf{u}), \ldots, f_m(\mathbf{u})\right), \\ &\text{s.t. } \mathbf{u} = (u_1, u_2, \ldots, u_n) \in U \subset R^n \\ &\qquad \mathbf{v} = (v_1, v_2, \ldots, v_m) \in V \subset R^m \end{aligned}$$

where $\mathbf{u}$ is the decision vector, U the decision space, $\mathbf{v}$ the objective vector and V the objective space. Due to the presence of multiple objective functions, a final solution of MOOP consists of a set of non-dominated solutions instead of a single point. The notion of *dominance* and related terms are defined below.

Definition 1. A decision vector $\mathbf{u_1}$ is said to dominate another solution $\mathbf{u_2}$ (denoted by $\mathbf{u_1} \prec \mathbf{u_2}$), if and only if

$$\forall i \in \{1, \ldots, m\} : f_i(\mathbf{u_1}) \leq f_i(\mathbf{u_2}) \wedge \exists k \in \{1, \ldots, m\} : f_k(\mathbf{u_1}) < f_k(\mathbf{u_2}). \quad (1)$$

Definition 2. A solution $\mathbf{u}$ is said to be non-dominated by any other solutions (a Pareto optimal solution) if and only if

$$\neg \exists \mathbf{u}^* \in U : \mathbf{u}^* \prec \mathbf{u}. \quad (2)$$

Definition 3. For a given MOOP, a Pareto optimal set in the decision space, $\mathcal{PS}$, is defined as

$$\mathcal{PS} := \{\mathbf{u} | \neg \exists \mathbf{u}^* \in U : \mathbf{u}^* \prec \mathbf{u}\}. \quad (3)$$

Definition 4. For a given MOOP, a Pareto optimal set in the objective space, $\mathcal{PF}$, is defined as

$$\mathcal{PF} := \{\mathbf{f}(\mathbf{u}) | \mathbf{u} \in \mathcal{PS}\}. \quad (4)$$

Since the size of Pareto optimal front, $\mathcal{PF}$ is infinite in general, which is impossible to obtain in practice, practical algorithms for MOOP yield a set of non-dominated solutions of a finite size. It should be noted that $\mathcal{PF}$ is always a non-dominated set by definition while a non-dominated set of solutions generated by an algorithm, which is denoted as a $\mathcal{GS}$, may not be a subset of $\mathcal{PF}$.

3. Description of conformational space annealing

Here, a new multiobjective optimization algorithm based on CSA is described. The CSA was initially developed to obtain the protein structure with the minimum potential energy, *i.e.*, to solve a single objective optimization problem. CSA has been successfully applied to various kinds of optimization problems with modification. The general framework of CSA is shown in Figure 1, and the description of MOCSA is given in Algorithm 1.

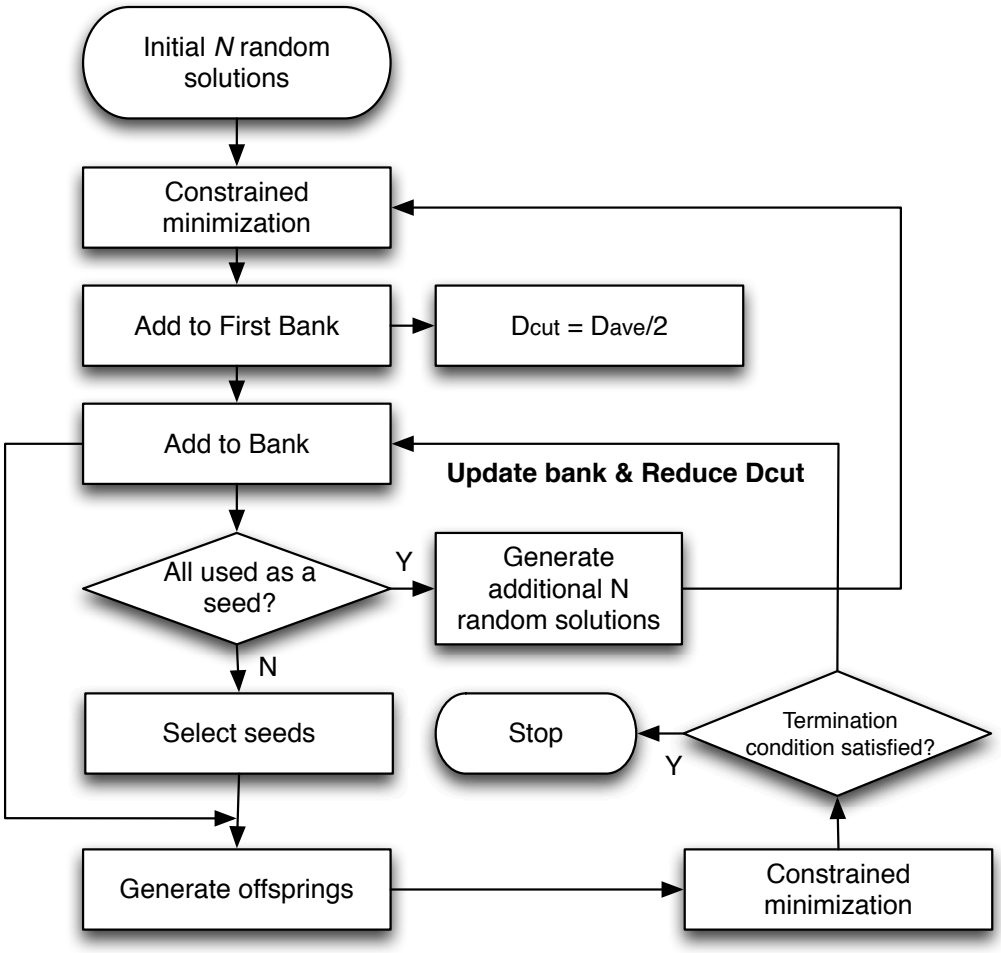

Figure 1: **Flow chart of CSA is shown.**

Algorithm 1 Multiobjective CSA algorithm

procedure MOCSA($N, N_s, N_c, N_m, G = 300, \alpha$)
    Initialize the bank, $\mathbb{P}$, with N random individuals
    Minimize($\mathbb{P}$) using a constrained local minimizer
    Initialize seed flags of all individuals to zeros : $s(i) \leftarrow 0, i = 1, \ldots, N$
    Get average distance, D_{ave}, between all pairs of individuals and set D_{cut} as $D_{ave}/2$: $D_{cut} \leftarrow D_{ave}/2$
    Initialize generation counter to zero : $g \leftarrow 0$
    Initialize the reserve bank, $\mathbb{R}$, to an empty set
    while $g < G$ **do**
        if $s(i) == 1 \quad \forall i$ **then**
            Generate N random individuals, $\mathbb{P}'$
            Minimize($\mathbb{P}'$)
            $\mathbb{P} \leftarrow \mathbb{P} \cap \mathbb{P}'$ ▷ Expand search space
        end if
        Evaluate Fitness of $\mathbb{P}$
        Select N_s seeds among individuals with $s(i) = 0$ and set $s(i)$ to 1
        $\mathbb{T}_c \leftarrow$ Generate $N_s N_c$ trial solutions by crossover
        $\mathbb{T}_m \leftarrow$ Generate $N_s N_m$ trial solutions by mutation
        $\mathbb{T} \leftarrow \mathbb{T}_c \cup \mathbb{T}_m$ ▷ Trial solutions
        if $G\%5 == 0$ **then**
            Minimize($\mathbb{T}$)
        end if
        Update($\mathbb{P}, \mathbb{T}, \mathbb{R}$)
        $D_{cut} \leftarrow max(\alpha D_{cut}, D_{ave}/5)$ ▷ Reduce D_{cut}
        $g \leftarrow g + 1$
    end while
end procedure

3.1. Conformational space annealing

CSA is a global optimization method which combines essential ingredients of three methods: Monte Carlo with minimization (MCM) (Li and Scheraga, 1987), genetic algorithm (GA) (Goldberg, 1989), and SA (Kirkpatrick et al., 1983). As in MCM, we consider only the solution/conformational space of local minima; in general, all solutions are minimized by a local minimizer. As in GA, we use a set of N solutions (called *bank* in CSA, denoted as $\mathbb{P}$) collectively, and we generate offsprings from the bank solutions by cross-over and mutation. Finally, as in SA, we introduce a distance parameter D_{cut}, which plays the role of the temperature in SA. In CSA, each solution is assumed to represent a hyper-sphere of radius D in the decision space. Diversity of sampling is directly controlled by introducing a distance measure between two solutions and comparing it with D_{cut}, to prevent two solutions from approaching too close to each other in the decision space. Similar to the manipulation of temperature in SA, the value of D_{cut} is initially set to a large value and is slowly reduced to a smaller value in CSA; hence the name conformational space annealing.

Compared to the conventional EA for multiobjective problems, MOCSA has three distinct features; (a) a ranking algorithm which considers the dominance relationship as well as the distance between solutions in the objective space, (b) an update rule with a dynamically varying distance cutoff measure to control the size of search space and to keep the diversity of sampling in the decision space and (c) the usage of a gradient-based constrained minimizer, feasible sequential quadratic programming (FSQP), for local search.

In CSA, we first initialize the *bank*, $\mathbb{P}$, with $N = 50$ random solutions which are subsequently minimized by FSQP constrained minimizer. The solutions in the bank are updated using subsequent solutions found during the course of optimization. The initial value of D_{cut} is set as $D_{avg}/2$, where D_{avg} is the average distance in the *decision* space between two solutions at the initial stage. A number of solutions (20 in this study) in the bank are selected as *seeds*. For each seed, 30 trial solutions are generated by cross-over between the seed and randomly chosen solutions from the bank. Additional 5 are generated by mutation of the seed. It should be noted that if a solution is used as a seed and not replaced by a offspring, it is excluded from the subsequent seed selection. The generated offsprings are locally minimized by FSQP which guarantees to improve a subset of objective functions without deteriorating the others and without violating given constraints. To limit the computational usage, the minimization is performed only once per

every five generation steps (see Algorithm 1).

Offsprings are used to update the bank, and detailed description on the updating rule is provided in Section 3.3. Once all solutions in the bank are used as seeds without generating better solutions, implying that the procedure might have reached a deadlock, we reset all bank solutions to be eligible for seeds again and repeat another round of the search procedure. After this additional search reaches a deadlock again, we expand our search space by adding additional 50 randomly generated and minimized solutions to the bank ($N = N + 50$), and repeat the whole procedure until a termination criterion is satisfied. The maximum number of generation is set to G. Typically, with $G = 300$ in this study, MOCSA is terminated before a deadlock occurs with the final bank size of $N = 50$.

3.2. Fitness function

For a given set of generated solutions, $\mathcal{GS}$, the fitness of solution i is evaluated in terms of n_i, m_i and d_i^{12}. n_i is the number of solutions in $\mathcal{GS}$ which dominate i. m_i is the number of solutions in $\mathcal{GS}$ dominated by i. d_i^{12} is the sum of distances from i to its nearest and second nearest neighbors in $\mathcal{GS}$ in the objective space. The relative fitness between two solutions, i and j, is determined by the comparing function shown in Algorithm 2. With a set of non-dominated solutions all values of n_i and m_i become zeros and the solution with the least value of d^{12} is considered as the worst.

3.3. Update rule

The solutions generated by crossover and mutation are locally minimized by FSQP constrained minimizer and we call them trial solutions. Each trial solution $\mathbf{t}$, is compared with the bank $\mathbb{P}$ for update procedure as shown in Algorithm 3. First, $\mathbf{w}$, the closest solution in $\mathbb{P}$ from $\mathbf{t}$ in the *decision* space is identified. If there exist dominated solutions in $\mathbb{P}$, the closest conformation search is performed only among them. Otherwise, $\mathbb{P}$ is a set of non-dominated solutions, and all in $\mathbb{P}$ are considered. Once $\mathbf{w}$ is found, the distance D in the decision space between $\mathbf{t}$ and $\mathbf{w}$ is calculated. If $D > D_{cut}$, the current cutoff distance, which indicates that $\mathbf{t}$ lies in a newly sampled region in the decision space, remote from the existing solutions in $\mathbb{P}$, the dominance relationship between $\mathbf{t}$ and the worst solution in $\mathbb{P}$, $\mathbf{u}$, is compared. If $D \leq D_{cut}$, $\mathbf{t}$ is compared with $\mathbf{w}$. The selection procedure, described in Section 3.4, is performed to determine which solution should be kept in $\mathbb{P}$. At each iteration step, D_{cut} is reduced with a pre-determined ratio, α. After D_{cut} reaches to its final value, $D_{avg}/5$, it is kept constant.

3.4. Selection procedure

In Algorithm 3, for a given trial solution $\mathbf{t}$ and a solution $\mathbf{x}$ in $\mathbb{P}$, $\mathbb{P}$ is updated as follows. If $\mathbf{t}$ dominates $\mathbf{x}$, $\mathbf{t}$ replaces $\mathbf{x}$. If $\mathbf{t}$ is dominated by $\mathbf{x}$, $\mathbf{t}$ is discarded and $\mathbf{x}$ stays in $\mathbb{P}$. If there is no dominance relationship between $\mathbf{t}$ and $\mathbf{x}$ and if $\mathbf{x}$ is better than $\mathbf{t}$ by Algorithm 2, $\mathbf{t}$ is discarded and $\mathbf{x}$ stays in $\mathbb{P}$. Finally, when $\mathbf{t}$ is better than $\mathbf{x}$ without dominance relationship between them, Algorithm 4 is used.

For the selection procedure, we introduce an additional set of non-dominated solutions, the reserve bank, $\mathbb{R}$. Due to the limited size of the bank, we may encounter a situation where a solution exists in $\mathbb{P}$ which is not dominated by the current bank, but dominated by a solution eliminated from the bank in an earlier generation. To solve this problem, non-dominated solutions eliminated from $\mathbb{P}$ are stored up to 500 in $\mathbb{R}$, which is conceptually similar to an *archive* in other EAs (Knowles and Corne, 2000; Zitzler et al., 2001). The difference is that $\mathbb{R}$ is used only when more than half of the solutions in $\mathbb{P}$ are non-dominated solutions because CSA focuses more on diverse sampling rather than optimization at the early stage of the optimization. Note that $\mathbb{R}$ keeps only non-dominated solutions.

4. Test suites

For the benchmark test of MOCSA, we have selected 12 widely used test problems in the field. They consist of ZDT (Zitzler et al., 2000) and DTLZ (Deb et al., 2002). Each test suite contains several functional forms and can feature various aspects of optimization algorithms. Comprehensive analysis on the characteristics of the two test suites are well documented by Huband *et al.* (Huband et al., 2006). In both suites, the input vector, $\mathbf{x}$, is divided into two sets $\mathbf{y}$ and $\mathbf{z}$ to construct test problems as follows,

$$\begin{aligned} \text{Given } \mathbf{x} &= \{x_1, \ldots, x_n\} \\ \text{let } \mathbf{y} &= \{y_1, \ldots, y_j\} = \{x_1, \ldots, x_j\} \\ \mathbf{z} &= \{z_1, \ldots, z_k\} = \{x_{j+1}, \ldots, x_n\} \end{aligned}$$

, where n and j are the dimensions of decision and objective spaces respectively and $k = n - j$.

4.1. ZDT

The ZDT problem suite consists of six test problems and is probably the most popular test suite to access multiobjective optimization algorithms.

Algorithm 2 Comparing function

function COMPARE(i, j)
    $n_{i(j)} \leftarrow$ Number of solutions in $\mathbb{P}$ which dominate **i**(**j**)
    $m_{i(j)} \leftarrow$ Number of solutions in $\mathbb{P}$ which are dominated by **i**(**j**)
    $d^{12}_{i(j)} \leftarrow$ The sum of distances from i to the nearest and second nearest neighbors in $\mathbb{P}$ in the *objective* space
    if $n_i < n_j$ **then**
        return i is better
    else if $n_i > n_j$ **then**
        return j is better
    else ▷ $n_i == n_j$
        if $m_i > m_j$ **then**
            return i is better
        else if $m_i < m_j$ **then**
            return j is better
        else ▷ $m_i == m_j$
            if $d^{12}_i > d^{12}_j$ **then**
                return i is better
            else
                return j is better
            end if
        end if
    end if
end function

Algorithm 3 Update procedure

```
1: procedure UPDATE(ℙ, 𝕋, ℝ)
2:     for t in 𝕋 do
3:         if all solutions in ℙ are non-dominated then
4:             w ← Nearest solution to t in ℙ in the decision space
5:         else
6:             w ← Nearest solution to t among dominated solutions in ℙ
       in the decision space
7:         end if
8:         if distance(t, w) > D_cut then    ▷ distance in the decision space
9:             u ← Worst solution in ℙ
10:            x ← u
11:        else
12:            x ← w
13:        end if
14:        if t ≺ x then                                  ▷ t dominates x
15:            t replaces x
16:        else if t ≻ x then                             ▷ x dominates t
17:            x stays in ℙ and t is discarded
18:        else if x is better than t by Algorithm 2 then
19:            x stays in ℙ and t is discarded
20:        else
21:            SELECT(t, x)
22:        end if
23:    end for
24: end procedure
```

Algorithm 4 Select procedure

procedure SELECT($\mathbf{t}, \mathbf{x}$) ▷ $\mathbf{t} \in \mathbb{T}$, $\mathbf{x} \in \mathbb{P}$, $\mathbf{t}$ is better than $\mathbf{x}$
  if $|\mathcal{ND}(\mathbb{P})| < |\mathbb{P}|/2$ or $\mathbf{x}$ is dominated by $\mathbf{x}' \in \mathbb{P}$ **then**
    $\mathbf{t}$ replaces $\mathbf{x}$ in $\mathbb{P}$
  else ▷ $\mathbf{x}$ is non-dominated in $\mathbb{P}$ and $\mathbb{R}$ is used
    if $\mathbf{t}$ is not dominated by $\mathbb{R}$ **then**
      $\mathbf{t}$ replaces $\mathbf{x}$ in $\mathbb{P}$
    else ▷ $\mathbf{t}$ is dominated by $\mathbb{R}$
      $\mathbf{u} \leftarrow$ Nearest dominating solution to $\mathbf{t}$ in $\mathbb{R}$ in the *objective* space
      Move $\mathbf{u}$ from $\mathbb{R}$ to $\mathbb{P}$
    end if
    Move $\mathbf{x}$ from $\mathbb{P}$ to $\mathbb{R}$
    $\mathbb{R} \leftarrow \mathcal{ND}(\mathbb{R})$
  end if
end procedure

The explicit functional forms of five ZDT problems are presented in Table 1. The ZDT test suite has two main advantages: (a) The Pareto fronts of the problems are known in exact forms and (b) benchmark results of many existing studies are available. However, there are shortcomings: (a) The problems have only two objectives, (b) none of the problems contain flat regions and (c) none of the problems have degenerate Pareto optimal front (Huband et al., 2006).

4.2. DTLZ

The original DTLZ suite consists of nine test problems which are scalable to any number of objectives. This scalability is important since it makes the test suite suitable for testing algorithms for *many* objective problems. The explicit functional forms of the first seven DTLZ problems are presented in Table 2. DTLZ8 and DTLZ9 are omitted in many benchmark studies due to their additional constraints and they are also omitted in this study.

5. Performance measures

To evaluate the performance of an algorithm, two aspects are considered: Solutions should be (a) as close to the Pareto front as possible and (b) as diversely distributed among them. Here, we have used four measures defined below.

Table 1: Five real-valued ZDT problems are described. The first objective depends only on the first decision variable as $f_1(y_1)$ and the second objective is given as $f_2(y_1, \mathbf{z}) = g(\mathbf{z})h(f_1(y_1), g(\mathbf{z}))$, where $\mathbf{y} = \{y_1, \ldots, y_j\} = \{x_1, \ldots, x_j\}$ and $\mathbf{z} = \{z_1, \ldots, z_k\} = \{x_{j+1}, \ldots, x_n\}$, where n and j are the dimensions of the decision space and the objective space. Unless the functional forms of f_1, g and h are separately given, they are identical to those of ZDT1.

Name	Problem	domains
ZDT1	$f_1 = y_1$ $g = 1 + 9\dot{\sum}_{i=1}^{k} z_i/k$ $h(f_1, g) = 1 - \sqrt{f_1/g}$	[0,1]
ZDT2	$h = 1 - (f_1/g)^2$	[0,1]
ZDT3	$h = 1 - \sqrt{f_1/g} - (f_1/g)sin(10\pi f_1)$	[0,1]
ZDT4	$g = 1 + 10k + \sum_{i=1}^{k}(z_i^2 - 10cos(4\pi z_i))$	$y_1 \in [0, 1]$ $z_1, \ldots, z_k \in [-5, 5]$
ZDT6	$f_1 = 1 - exp(-4y_1)sin^6(6\pi y_1)$ $g = 1 + 9(\dot{\sum_{i=1}^{k}} z_i/k))^{0.25}$ $h = 1 - (f_1/g)^2$	[0,1]

Table 2: Seven real-valued DTLZ test problems are described. The objective space is j-dimensional and j is set to 3 for benchmarking in this study. The input vectors are $\mathbf{y} = \{y_1, \ldots, y_j\} = \{x_1, \ldots, x_j\}$ and $\mathbf{z} = \{z_1, \ldots, z_k\} = \{x_{j+1}, \ldots, x_n\}$, where n is the dimension of decision space and $k = n - j$.

Name	Problem	domains
DTLZ1	$f_1 = \frac{1}{2}(1+g)\prod_{i=1}^{j-1} y_i$ $f_{m=2:j-1} = \frac{1}{2}(1+g)\left(\prod_{i=1}^{j-m} y_i\right)(1-y_{j-m+1})$ $f_j = \frac{1}{2}(1+g)(1-y_1)$ $g = 100[k + \sum_{i=1}^{k}\left((z_i-0.5)^2 - cos(20\pi(z_i-0.5))\right)]$	[0,1]
DTLZ2	$f_1 = (1+g)\prod_{i=1}^{j-1} cos(\pi y_i/2)$ $f_{m=2:j-1} = (1+g)\left(\prod_{i=1}^{j-m} cos(\pi y_i/2)\right) sin(\pi y_{j-m+1}/2)$ $f_j = (1+g)sin(\pi y_1/2)$ $g = \sum_{i=1}^{k}(z_i-0.5)^2$	[0,1]
DTLZ3	Same as DTLZ2, except g is replaced by the one from DTLZ1	[0,1]
DTLZ4	Same as DTLZ2, except y_i are replaced by $y_i^{0.5}$	[0,1]
DTLZ5	Same as DTLZ2, except $y_{2,\ldots,j-1}$ are replaced by $\frac{1+2gy_i}{2(1+g)}$.	[0,1]
DTLZ6	Same as DTLZ5, except g is replaced by $g = \sum_{i=1}^{k} z_i^{0.1}$.	[0,1]
DTLZ7	$f_{m=1:j-1} = y_m$ $f_j = (1+g)\left(j - \sum_{i=1}^{j-1}[\frac{f_i}{1+g}(1+sin(3\pi f_i))]\right)$ $g = 1 + 9\sum_{i=1}^{k} z_i/k$ [13]	[0,1]

5.1. Spacing (S) and average distance between solutions ($\langle d \rangle$)

The spacing, S, measures how uniformly generated solutions $\mathcal{GS}$ are distributed in the objective space and it is defined as

$$S = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - \langle d \rangle)^2},$$

$$\text{where} \quad d_i = \min_{\substack{j \in \mathcal{GS} \\ j \neq i}} \sum_{k=1}^{m} |f_k^i - f_k^j|,$$

$$\langle d \rangle = \sum_{i=1}^{N} d_i / N$$

, where N is the number of solutions in $\mathcal{GS}$, d_i is the distance between the solution i and its nearest neighbor in the objective space in terms of the Hamming distance and $\langle d \rangle$ is the average of d_i (Scott, 1995). Ideally, we want a larger value of $\langle d \rangle$ with a smaller value of S.

5.2. Generational distance (GD)

The generational distance measures the average distance between the generated solutions ($\mathcal{GS}$) and the Pareto front (Van Veldhuizen and Lamont, 2000). GD is defined as

$$GD = \frac{1}{N}\left(\sum_{i=1}^{N} l_i^2\right)^{1/2} \tag{5}$$

, where N is the number of solutions in $\mathcal{GS}$ and l_i is the Euclidean distance between the solution $i \in \mathcal{GS}$ and its nearest in $\mathcal{PF}$. If all solutions in $\mathcal{GS}$ lie on $\mathcal{PF}$, GD is 0. Therefore, a lower GD value is preferred.

5.3. Error ratio (ER)

The error ratio measures the fraction of solutions which are not on $\mathcal{PF}$ and it is defined as

$$ER = \frac{\sum_{i=1}^{N} e_i}{N} \tag{6}$$

, where N is the number of generated solutions and $e_i = 0$ if the solution i belongs to $\mathcal{PF}$ within 0.01, $e_i = 1$ otherwise (Van Veldhuizen and Lamont, 2000).

Table 3: Dimensions of decision and objective space of benchmark problems are shown.

Problem	dimension of decision space	dimension of objective space
ZDT1-3	30	2
ZDT4,6	10	2
DTLZ1-7	10	3

6. Result

The numerical results on 12 test problems obtained by MOCSA and NSGA2 are shown in Table 4 and Figures 2 & 3. For benchmarking test, the number of population and trial solutions were set to 50 and 700 for both methods. The NSGA2 was iterated for 250,000 generations and probabilities of crossover and mutation were fixed at 0.9 and 1/24. For MOCSA, the number of seeds used to generate trial solutions was set to 20. The dimensions of decision and objective space of benchmark problems are listed in Table 3.

For ZDT problems, MOCSA outperforms NSGA2 in terms of both convergence and diversity of the solutions, except ZDT6. For ZDT6, MOCSA is better than NSGA2 by $\langle d \rangle$ but worse by S. For all ZDT problems, entire solutions obtained by MOCSA lies within 0.01 on the Pareto front, shown in red circles in Figure 2, while NSGA2 solutions are slightly off as shown in Table 4. Regarding the spread of the solutions, MOCSA results are more uniformly distributed on the Pareto front, signified by lower S values, than NSGA2 for the first four ZDT problems.

For DTLZ problems, MOCSA outperforms NSGA2 in almost all aspects. For DTLZ7, 8% of solutions by MOCSA is not on the Pareto front, while 22% is the case for NSGA2. In terms of S and $\langle d \rangle$, MOCSA provides more evenly distributed solutions, covering a wider range of the objective space. It should be noted that $\langle d \rangle$ by MOCSA is larger than that by NSGA2 for for all 12 problems tested (by the factor of 1.59 on average), which is signified by the fact that red circles by MOCSA cover Pareto fronts evenly while blue crosses by NSGA2 appear to be locally and unevenly clustered in Figures 2 & 3.

7. Conclusion

In this paper, we have introduced a novel multiobjective optimization algorithm by using the conformational space annealing (CSA) algorithm,

Table 4: Four performance measures are shown for MOCSA and NSGA2

Problem	MOCSA				NSGA2			
	$\langle d \rangle$	S	GD	ER	$\langle d \rangle$	S	GD	ER
ZDT1	0.0404	0.0055	0.0000	0	0.0270	0.0156	0.0011	0.04
ZDT2	0.0404	0.0082	0.0000	0	0.0292	0.0146	0.0212	0.02
ZDT3	0.0438	0.0148	0.0001	0	0.0329	0.0201	0.0020	0.02
ZDT4	0.0404	0.0097	0.0000	0	0.0328	0.0159	0.0006	0.02
ZDT6	0.0327	0.0150	0.0000	0	0.0216	0.0119	0.0000	0
DTLZ1	0.1114	0.0068	0.0000	0	0.0615	0.0319	0.0000	0
DTLZ2	0.2319	0.0646	0.0021	0.02	0.1361	0.0683	0.0020	0.04
DTLZ3	0.2770	0.0225	0.0000	0	0.1139	0.0739	0.0000	0
DTLZ4	0.2478	0.0424	0.0009	0	0.1630	0.0898	0.0019	0.02
DTLZ5	0.0487	0.0059	0.0000	0	0.0309	0.0176	0.0610	0.06
DTLZ6	0.0484	0.0156	0.0000	0	0.0306	0.0135	0.0000	0
DTLZ7	0.2897	0.0510	0.0011	0.04	0.1880	0.1322	0.0071	0.22

MOCSA. Benchmark results on 12 test problems show that MOCSA finds better solutions than NSGA2, in terms of four criteria tested. Solutions by MOCSA are closer to the Pareto front, a higher fraction of them are on the Pareto front, they cover a wider objective space, and they are more evenly distributed on average. We note that the efficiency of MOCSA arises from the fact that it controls the diversity of solutions in the decision space as well as in the objective space.

8. Acknowledgement

The authors acknowledge support from Creative Research Initiatives (Center for In Silico Protein Science, 20110000040) of MEST/KOSEF. We thank Korea Institute for Advanced Study for providing computing resources (KIAS Center for Advanced Computation Linux Cluster) for this work. We also would like to acknowledge the support from the KISTI Supercomputing Center (KSC-2012-C3-02).

References

Barán, B., Schaerer, M., 2003. A multiobjective ant colony system for vehicle routing problem with time windows. In: IASTED International Multi-Conference on Applied Informatics. No. 21. pp. 97–102.

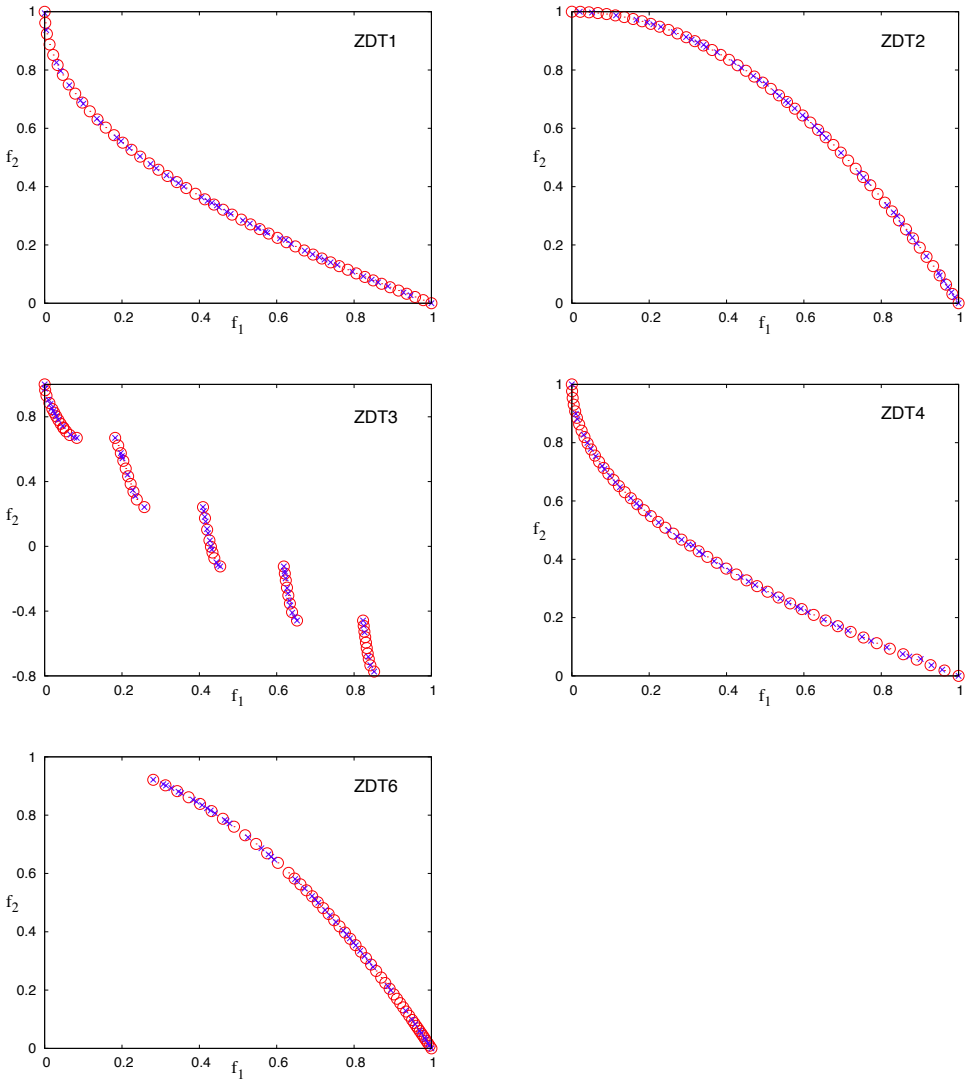

Figure 2: **Benchmark results on five ZDT problems are shown for MOCSA (red circle) and NSGA2 (blue X)**

Coello Coello, C., Becerra, R., 2003. Evolutionary multiobjective optimization using a cultural algorithm. In: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE. IEEE, pp. 6–13.

Coello Coello, C., Cortés, N., 2002. An approach to solve multiobjective optimization problems based on an artificial immune system. In: Proceedings of First International Conference on Artificial Immune Systems 2002; Canterbury, UK. pp. 212–221.

Coello Coello, C., Lamont, G., Van Veldhuizen, D., 2007. Evolutionary algorithms for solving multi-objective problems. Vol. 5. Springer-Verlag New York Inc.

Coello Coello, C., Lechuga, M., 2002. Mopso: A proposal for multiple objective particle swarm optimization. In: Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on. Vol. 2. IEEE, pp. 1051–1056.

Coello Coello, C., Toscano Pulido, G., 2001. A micro-genetic algorithm for multiobjective optimization. In: Evolutionary Multi-Criterion Optimization. Springer, pp. 126–140.

Deb, K., 2001. Multi-objective optimization using evolutionary algorithms. Vol. 16. Wiley.

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. Lecture notes in computer science 1917, 849–858.

Deb, K., Thiele, L., Laumanns, M., Zitzler, E., 2002. Scalable multi-objective optimization test problems. In: Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA). Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA), pp. 825–830.

Doerner, K., Gutjahr, W., Hartl, R., Strauss, C., Stummer, C., 2004. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. Annals of Operations Research 131 (1), 79–99.

Gandibleux, X., Mezdaoui, N., Fréville, A., 1997. A tabu search procedure to solve multiobjective combinatorial optimization problem. Lecture notes in economics and mathematical systems, 291–300.

Gao, J., Wang, J., 2011. A hybrid quantum-inspired immune algorithm for multiobjective optimization. Applied Mathematics and Computation 217 (9), 4754–4770.

Goldberg, D., 1989. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional.

Hansen, M., 1997. Tabu search for multiobjective optimization: Mots. In: Proceedings of the 13th International Conference on Multiple Criteria Decision Making (MCDM97), Cape Town, South Africa. Citeseer, pp. 574–586.

Horn, J., Nafpliotis, N., Goldberg, D., 1994. A niched pareto genetic algorithm for multiobjective optimization. In: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on. Ieee, pp. 82–87.

Huband, S., Hingston, P., Barone, L., While, L., 2006. A review of multiobjective test problems and a scalable test problem toolkit. Evolutionary Computation, IEEE Transactions on 10 (5), 477–506.

Joo, K., Lee, J., Kim, I., Lee, S., Lee, J., 2008. Multiple sequence alignment by conformational space annealing. Biophysical journal 95 (10), 4813–4819.

Joo, K., Lee, J., Seo, J., Lee, K., Kim, B., Lee, J., 2009. All-atom chain-building by optimizing modeller energy function using conformational space annealing. Proteins: Structure, Function, and Bioinformatics 75 (4), 1010–1023.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.

Knowles, J., Corne, D., 2000. M-paes: A memetic algorithm for multiobjective optimization. In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on. Vol. 1. IEEE, pp. 325–332.

Lee, J., Gross, S. P., Lee, J., May 2012. Modularity optimization by conformational space annealing. Phys. Rev. E 85, 056702.

Lee, J., Lee, I.-H., Lee, J., Aug 2003. Unbiased global optimization of lennard-jones clusters for $N \leq 201$ using the conformational space annealing method. Phys. Rev. Lett. 91, 080201.

Lee, J., Liwo, A., Ripoll, D., Pillardy, J., Scheraga, H., Jan. 1999. Calculation of protein conformation by global optimization of a potential energy function. Proteins Structure Function and Genetics 37 (s 3), 204–208.

Li, Z., Scheraga, H. A., 1987. Monte carlo-minimization approach to the multiple-minima

problem in protein folding. Proceedings of the National Academy of Sciences 84 (19), 6611–6615.

Liwo, A., Lee, J., Ripoll, D. R., Pillardy, J., Scheraga, H., May 1999. Protein structure prediction by global optimization of a potential energy function. Proceedings of the National Academy of Sciences of the United States of America 96 (10), 5482–5.

Luh, G., Chueh, C., Liu, W., 2003. Moia: multi-objective immune algorithm. Engineering Optimization 35 (2), 143–164.

Nam, D., Park, C., 2000. Multiobjective simulated annealing: A comparative study to evolutionary algorithms. International Journal of Fuzzy Systems 2 (2), 87–97.

Parsopoulos, K., Vrahatis, M., 2002. Particle swarm optimization method in multiobjective problems. In: Proceedings of the 2002 ACM symposium on Applied computing. ACM, pp. 603–607.

Pillardy, J., Czaplewski, C., Liwo, A., Lee, J., Ripoll, D. R., Kaźmierkiewicz, R., Oldziej, S., Wedemeyer, W. J., Gibson, K. D., Arnautova, Y. a., Saunders, J., Ye, Y. J., Scheraga, H., Feb. 2001. Recent improvements in prediction of protein structure by global optimization of a potential energy function. Proceedings of the National Academy of Sciences of the United States of America 98 (5), 2329–33.

Schaffer, J., 1985. Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the 1st international Conference on Genetic Algorithms. L. Erlbaum Associates Inc., pp. 93–100.

Scott, J., 1995. Fault tolerant design using single and multi-criteria genetic algorithms. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology.

Suman, B., Kumar, P., 2005. A survey of simulated annealing as a tool for single and multiobjective optimization. Journal of the Operational Research Society 57 (10), 1143–1160.

Van Veldhuizen, D., Lamont, G., 2000. On measuring multiobjective evolutionary algorithm performance. In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on. Vol. 1. IEEE, pp. 204–211.

Zitzler, E., Deb, K., Thiele, L., 2000. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation 8 (2), 173–195.

Zitzler, E., Laumanns, M., Thiele, L., Zitzler, E., Zitzler, E., Thiele, L., Thiele, L., 2001. Spea2: Improving the strength pareto evolutionary algorithm.

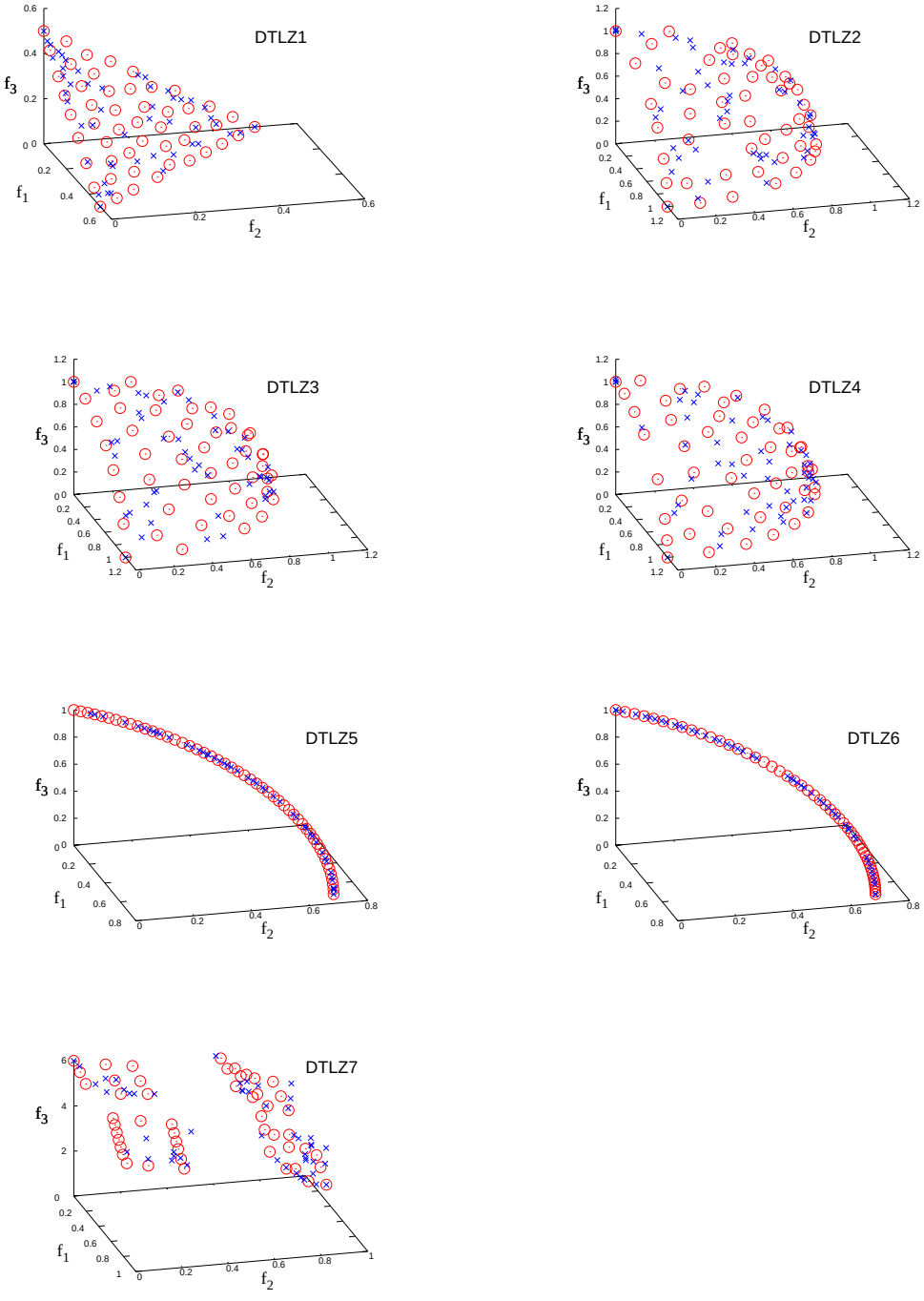

Figure 3: **Benchmark results on seven DTLZ test problems are shown for MOCSA (red circle) and NSGA2 (blue X)**